TRANSCRIBING POLYPHONIC PIANO MUSIC USING A NEURAL NETWORK

Kyle Newman
University of Victoria
V00781162

Second Author
Retain these fake authors in submission to preserve the formatting

Third Author
Affiliation3
author3@ismir.edu

ABSTRACT

In this report, a study is done of the techniques used for Automatic Music Transcription. In particular, this report studies the techniques that relate to Dense Neural Networks and Convolutional Neural Networks. Next, a framework is developed using a Dense Neural Network. Experiments are performed using a subset of the Saarland Music Data (SMD) dataset. This report observes the results from tuning the hyper-parameters and optimizers so as to maximize the framework's effect.

1. INTRODUCTION

Programming a computer to transcribe audio is an ongoing a challenge in Music Information Retrieval. While Musicians practise transcription with great success, the structure of music is difficult for a computer to properly model. Furthermore, computers don't share a human being's ability to separate and identify overlapping sounds. Thus, the Western world's characteristic polyphonic style is particularly challenging for computers.

In consideration of Automatic Music Transcription (AMT), there are two main obstacles. The most immediate problem is identifying the pitches in a piece of polyphonic music. In order to identify these pitches, we must examine the frequency domain and identify fundamental frequencies we can use to classify these pitches. However, in western music, a composition may be performed with one or more instruments – with each having its own characteristic timbre. This timbre means that each instrument expresses an arbitrary pitch by means of a unique spectrum respective to other instruments. Thus, if we want to classify an audio input by identifying basis spectra (pitches) in its frequency representation, the approach must be capable of recognizing the spectra of a given pitch with minimal ambiguities, and do so for many possible representations of the same pitch. Otherwise our attempt at transcription will be restricted even by the instruments we wish to transcribe.

One of the most powerful and unique features in the western style of music is polyphony. In this style, music is not only expressed through a range of sounds, but through a mix of ensembles. Polyphony is a style which simultaneously combines numerous parts, each forming an individual melody and harmonizing with each other. This is a powerful tool, and leads to a significant range of forms and techniques. Polyphony allows for a greater range of expression, and does so by increasing the range of pitch combinations available to the composer (sometimes called the output space [19]).

Polyphony presents a particular problem for AMT, as the overlapping of voices means that the basis spectra for the pitches may overlap in the frequency spectrum. Since these basis spectra are not easy to classify in isolation, the overlapping of spectra lead to further ambiguities at the classification stage.

How can one resolve the ambiguities of overlapping pitches? Music is typically composed with some kind of expressive meaning. The sequence, combination and repetition of pitches leads the listener to derive meaning in its pattern. In information theory, this is called a Temporal Structure [19] as meaning is derived from the sequence of pitches over time. Music shares this structure with language, as a listener cannot comprehend a sentence until every word has been spoken. However, an intelligent listener can predict the point the speaker is making based on what has been said thus far. Human beings are able to model language. Thus, by using a language model, we can predict the next word in a sequence and derive meaning ahead of time.

In fact, the problems that speech recognition address are similar to those in Automatic Music Transcription. Consider an example sentence: *I want to go there*. In order to classify each word in the sentence, we must identify the basis spectra of each word. However, the words in a language can be ambiguous. I and eye are homophones, and so are there, their and they're. It would stand to reason that their basis spectra are nearly identical. Therefore, to resolve these ambiguities, one must consider their place in the sequence. I is far more likely to begin a sentence than eye, and there is more likely to immediately follow go than their or they're. Speech recognition techniques use hybrid

models to both classify possible words, and fit them into sentences [1].

In [19], Sigtia proposes a hybrid Neural Network model inspired by the techniques used in speech recognition. We have two models, the Acoustic Model (AM) which takes the audio and identifies note candidates, and a Music Language Model (MLM) which determines the most likely choices given sequence of inputs up to this point. This architecture uses a Convolutional Neural Network for its Acoustic Model and a Recurrent Neural Network for its Music Language Model.

2. NEURAL NETWORKS

A Neural Network (NN) is a structure that predicts some output y from some input x. In a supervised structure, a Neural Network accomplishes this prediction using a prepared set of data called a *dataset*. The Neural Network is fed a series of input parameters along with their expected outputs and the NN is expected to learn which parameters best allow it to approximate an output based on any input.

Neural Networks were originally inspired by the neurons and synapses in the human brain [12]. For a Neural Network, we treat these synapses as constant weights and their intersection at a given layer as neurons. If we represent these synapses as a linear combination of weights, a layer can be more simply expressed as a weight matrix W with an added bias b, where each parameter in the matrix represents a synapse to the Neuron. By transforming the input parameters along multiple layers, we can directly transform the input into its corresponding output given the right weights in each W using the equation.

$$z = W \cdot x + b$$
$$a = \sigma(z)$$

Where z is the computation at a given layer and a measures the activation of z. For example, the sigmoid function

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

maps the input z to the range (0,1), where a higher score indicates a higher activation at that neuron.

A Neural Network will learn the correct weights through an algorithm called Back Propogation. First, we quantify how wrong the predicted output is by creating a cost function. Our goal is to minimize the cost which in turn reduces the difference between the NN's output and the true output. In order to minimize the cost, we must find the global minimum of the cost function, therefore we apply partial derivation iteratively in order to decide in which direction to travel, eventually settling at the global minimum. This is called Gradient Descent. By finding the minimum cost, we determine the parameter weights which best transform the input.

The classic Neural Network is effective for classification and regression, however it isn't well suited for sequential data. If we intend to use a Neural Network for our MLM, we need something that will consider transformations in the context of previous inputs. The Recurrent Neural Network does just this. Instead of a simple matrix transformation at each layer, have an additional recurrent matrix which transforms the data according to previous inputs. This connection allows an RNN to better model sequential data, making it suited towards temporal structures like speech and music.

The third type of Neural Network we want to consider is the Convolutional Neural Network (ConvNet). Instead of simple matrix transformations, ConvNets classify features using a series of feature maps, pooling layers, and fully connected layers which extract and classify features directly from the signal. ConvNets were inspired by the human retina [6], and were designed for image processing. Each layer is trained to recognize a particular feature in the input and extract it, thus turning an input signal into a stack of feature signals.

3. RELATED WORKS

While there are many approaches to Automatic Music Transcription, Convolutional Neural Networks are a relatively new approach. There are two main unsupervised approaches for music transcription: spectrogram factorization techniques, and discriminative techniques. Spectrogram factorization techniques aim to transform an audio input into a combination of basis spectra (pitches). The central goal of this approach is to determine an efficient method of describing this basis spectra and is typically done by learning a dictionary of spectra related to these pitches [12]. However, as these bases don't necessarily correspond to the musical pitches exactly, these techniques are often limited in their musical scope as harmonic and polyphonic limits need to be set in advance.

3.1 Spectrogram Factorization

The earliest methods for Spectrogram factorization use non-negative matrix factorization as their central component. Work in this field started as early as 2003 with Smaragdis' paper [16]. Non-negative matrix factorization is the process of describing an input matrix X as the product of two matrices W and H. In this problem, the input matrix X is modelled as the DFT of an input waveform modelled over time. The columns of X represent the frequency bins, and the rows represent time frames. Since the magnitude spectrum of audio is strictly positive, we can employ non-negative matrix factorization to derive two matrices.

The matrices W and H each contain a dimension of X, along with a dimension given prior to the factorization. Thus these matrices summarize the horizontal and vertical structure respectively of X, fitted into rows of rank R. For example, if the rank was 3, then H would be a matrix which contains three rows of time information, while W would contain 3 rows of frequency information.

While this approach is quite simple, it's main shortcoming is that it requires a specific harmonic profile. If multiple timbres where introduced, the techniques ability to factorize based on pitch spectra would be hampered. Some noteable efforts have been made in this approach. More advanced techniques include the Probabilistic Latent Variable Component Analysis (PLCA) [19] which adds a probabilistic model on to the spectrogram factorization.

3.2 Discriminative Approach

The second approach involves discriminative techniques. This approach revolves around directly classifying features in the input audio. Neural Networks like the one described in this report fit into this category. This approach is useful as the harmonic limitations observed in spectrogram factorization techniques are mitigated [12]. Rather than build a model that describes a limited range of instrument-specific pitches, discriminative models can be trained with a large quantity of information to create complicated classifiers. These classifiers are not constrained to a single timbre, thus the classification of pitches can be generalized without worrying about harmonic or polyphonic constraints.

Early techniques include Support Vector Machines (SVM)[9], and Deep Belief Networks (DBN). In the discriminative approach, classifiers are usually trained on a set of training and testing sets, where each set contains a sample of audio. Audio is preprocessed using some form of Fourier Transform and windowing in order to achieve successive frames of the inputs frequency spectrum. The audio is then prepared through either normalization or standardization. Finally, classification is done in each frame in the input.

3.3 Convolution Networks in Music Information Retrieval

The use of Convolutional Neural Networks originated in image processing, however in recent years they have been used in speech recognition tasks with great success [1]. In the case of Automatic Music Transcription, they've remained largely unexplored [19].

While ConvNets haven't been used in AMT, it has been used with success in source separation problems. Simpson's article [20] describes a method separating the vocal and accompaniment features in a piece of pop music.

Simpson's paper uses Convolutional Neural Networks to tackle a famous problem in signal processing called the "Cocktail Party Problem" in which the engineer attempts to distinguish separate voices from a single audio source. This particular problem highlights the ConvNet's effectiveness at classifying spectral features in a noisy waveform, as the Networks must recognize a particular timbre from the overall mix and extract it from the whole. In fact, Simpson particularly notes this feature in his conclusion.

4. IMPLEMENTATION

4.1 The Dataset

While initial work was done with basic data (notes, scales), the final dataset is a subset taken from the Saarland Music Data (SMD) [15] dataset. This dataset provides audio-midi pairs of student performances recorded on the Yamaha Disklavier using a stereo microphone pair. The recordings were then converted into audio-midi pairs using Steinberg Cubase 4. The data is provided in MP3 format, and were thus converted back into WAV format using Audacity.

There were two issues of inconsistency in the pairs. The first is that the WAV files are noticeably longer than their corresponding midi counterparts. When looking at the WAV files in log amplitude, it was observed that there is some silence or fade out left in the tail of each WAV file that wasn't encoded in the resulting midi. Thus, the final WAV files are truncated such that the pairs are equal in number of samples. The second inconsistency is noted on the Saarland website. The Midi parser introduces a global offset in pitches. Encoding the WAV into midi personally provided similar results. Attempts to mitigate this were unsuccessful thus a subset of the dataset was selected. Only pairs whose offset was strictly less than 0.6 seconds were considered. Thus the final dataset is only around 67147 frames.

4.2 Preprocessing

The processed representation for the audio input is done by sampling the waveform to generate a frequency representation. Typically this is done with a Short Time Fourier Transform (STFT) which creates a spectrogram of linearly spaced frequency bins. However the STFT is not as useful in music as it is in other applications. In music, notes are divided into octaves. An octave is defined as being twice the frequency of its base, which means octave-based frequency bands are logarithmically spaced. If the musical waveform were to be represented using linear spacing, information would be lost in the higher frequencies.

The Constant Q-Transform (CQT) is a frequency representation with logarithmic spaced bins. Furthermore, the CQT is designed to space bins across octaves rather than by arbitrary frequencies. For example, if we specify 12 bins per octave across 7 octaves, we would have 84 bins

at 1 bin per semitone. The CQT also has the advantage of being lower dimensional while still maintaining high resolution. In [19], the CQT has 252 bins, while in [20] the Short Time Fourier Transform has 1025 bins. This smaller bin count translates to a much lower dimensionality for the input to the Neural Network.

The audio is downsampled from 44.1 kHz to 16 kHz, then the CQT is computed over 7 octaves at 36 bins per octave. This gives 252 bins at 3 bins per semitone, and a frame rate of 31.25 frames per second.

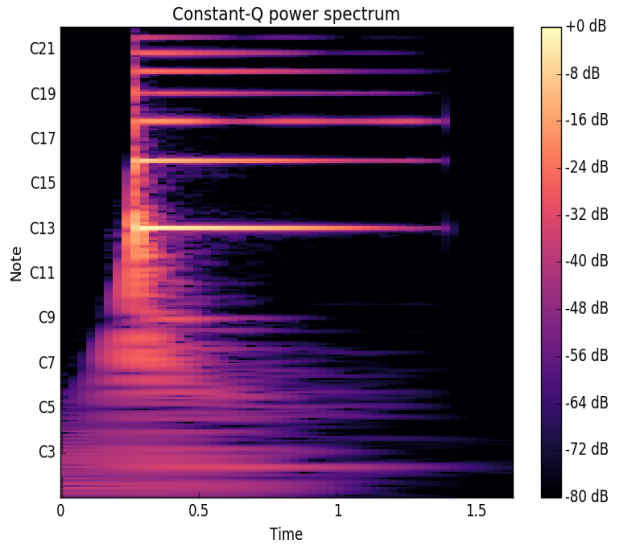


Figure 1: CQT for the note A4

As an initial test, the magnitudes are directly mapped in the CQT to an 88 dimensional binary output using constant thresholding. In this method, a threshold is assigned based on a fraction of the maximum value in the array. For every magnitude which is greater than the threshold, a 1 is assigned to the corresponding feature in the output vector. All other values are set to 0. We can convert this binary output to midi trivially as semitones and midi note values are both indexed.

For transcription problems, we wish to represent midi using an array as we are trying to classify only the presence of features in the input data. This can be accomplished using piano roll representations. Since we only care about the presence of features, we threshold the velocity information into a binary representation. While converting midi into a binary representation is non-trivial, some libraries exist that perform the conversion [5]. However, in [5], the piano representation uses a 128 dimensional vector which corresponds to midi note numbers. Since we are classifying piano notes rather than midi notes, we shift our piano roll into an 88 dimensional array so as to lower the dimensionality of the output layer in the neural network.

4.3 Training Model

The training model being tested is a Deep Neural Network (DNN). The model has 252 units in the input layer corresponding to the 252 bins in the CQT. The output layer is 88 units, corresponding to the 88 notes on the piano. This report mostly follows [19] in its process, so we use start with their recommended hyper-parameters, then experiment for optimal results. We test the number of hidden layers L $\in \{1, 2\}$, we fix the number of hidden units to be 125, and test two different activation functions act $\in \{ReLU, Sigmoid\}$. Mini Batches were fixed to a size of 100. Dropout is recommended, but we test both with and without it. We test the optimization of our Neural Network using SGD, ADADELTA [23], Adam, Adamax[12] and Nadam. Finally, we use early stopping to prevent overfitting setting it to monitor value loss with a patience of 20 epochs.

There are five optimizers considered for this model. The standard optimizer is Stochastic Gradient Descent (SGD), which uses gradient descent to minize the cost between the output and the ground truth. While it is effective, more advanced options were considered. Adadelta is used in [17] as it is an optimizer that changes its learning rate over iterations. Adam is an optimizer that combines the advantages of AdaGrad and RMSProp. It is an adaptive algorithm with little memory requirement. Adamax is an extension on Adam based on the infinity norm. Nadam is simply Adam with nesterov momentum.

4.4 Postprocessing

Neural Networks that rely on sigmoid activation will have outputs on a range between 0 and 1, however we want binary outputs. We use constant thresholding again to choose the features with the highest activation. If a feature is within a fraction of the maximum activation, it is set to 1 and all others are set to 0.

4.5 Metrics

Measurements are done on a cell by cell basis. For every bin in our 88 dimensional output vector, we compare with a representation of our ground-truth midi value. If the bins match, we call that a True Positive (TP). If the output has a 1 and the ground-truth is 0, we call that a False Positive (FP). If the output has a 0 and the ground-truth a 1, that is a False Negative (FN). For evaluation, we use the precision, recall, accuracy, and F-measure.

$$Precision : \sum_{n=1}^{N} \frac{TP[t]}{TP[t] + FP[t]}$$

$$Recall : \sum_{n=1}^{N} \frac{TP[t]}{TP[t] + FN[t]}$$

$$Accuracy : \sum_{n=1}^{N} \frac{TP[t]}{TP[t] + FP[t] + FN[t]}.$$

$$F - Measure : \frac{2 * Precision * Recall}{Precision + Recall}.$$

Where precision is a a metric of how many *selected* features are relevant, accuracy determines how many *relevant* features are selected, accuracy is the percentage of correct features as a fraction of the whole. The fmeasure is the most useful of the metrics for multi-labal classification, as precision can conceivably derive a perfect score from assigning a value at every feature. The fmeasure is the weighted harmonic mean of precision and recall, and therefore provides a middle ground between scoring for successful assignments (precision) and unsuccessful assignments (recall). All metrics are calculated on a range of 0 to 1. Where 1 is the highest score.

5. RESULTS

The intent of this project was to use a Neural Network to transcribe polyphonic piano music. When building this project, the goal was to approach the results in [19] as closely as possible. The results were largely successful as the results were indeed quite similar.

For the hyper-paramaters, there were some differences. For hidden layers, we tested using either one or two hidden layers. We found the number of hidden layers to have little impact on the score. Whether using two hidden layers or one, the output was largely the same. Perhaps this has to do with constraining the hidden units (both layers were fixed at 125 units). The ReLU activation wasn't very effective for our purposes, as the output scored poorly. We used the sigmoid function for all layers. Dropout was recommended, as it is supposed to prevent overfitting. However, concerning score it provided sub-optimal results when used on every layer. Training the Neural Network with Dropout while using Nadam results in an fmeasure of 0.56, with one layer of dropout we get an fmeasure score of 0.65 and without dropout at all we would get 0.72. Thus, there is a significant difference in score in using dropout. On observation, training the Neural Network without dropout seems to indeed suffer from overfitting as the results seem to be too dense. The fmeasure score is supposed to score against overfitting, however this doesn't seem to be reflected in the metrics. Applying dropout on only the hidden layer seemed to provide the best compromise.

Once these hyper-paramaters were fixed, we tested on the five optimization algorithms. As expected, the advanced algorithms provided substantial benefits over the basic Stochastic Gradient Descent.

Input Layer = 252,
Hidden Layer$_1$ = 125,
Dropout=0.3,
Hidden Layer$_2$ = 125,
Output Layer = 88,
Hidden Layers = 2,
act='sigmoid',
early stopping,

Table 1: The parameters for the final prototype.

Optimizer Results				
Optimzers	Precision	Recall	Accuracy	Fmeasure
SGD	0.1059	0.4447	0.0138	0.1709
AdaDelta	**0.6693**	0.5115	0.2868	0.5789
Adam	0.6499	0.6426	**0.2997**	0.6456
Adamax	0.6495	0.6405	0.2926	0.6441
Nadam	0.6352	**0.6663**	0.2940	**0.6496**

Table 2: Optimizers and their scores from the training model.

Of the optimization algorithms considered, Nadam scored the highest in both recall and fmeasure, while AdaDelta had the highest precision. Adam and the two extensions scored quite similarly, however with early stopping there were very noticeable differences in when each algorithm peaked. The epoch numbers are provided in table 3:

Thus while each Adam algorithm performed similarly, Nadam reached its score the fastest before peaking, while Adamax continued until the maximum number of epochs was reached.

The main loss of accuracy in the framework is in post processing. In this project, post processing is done using

Optimzers	Epochs
SGD	23/200
AdaDelta	173/200
Adam	111/200
Adamax	200/200
Nadam	**64/200**

Table 3: Number of Epochs for each Optimizer before Early Stopping.

Precision	Recall	Accuracy	Fmeasure
0.2579	0.1927	0.1352	0.1986

Table 4: Metrics obtained from the test set.

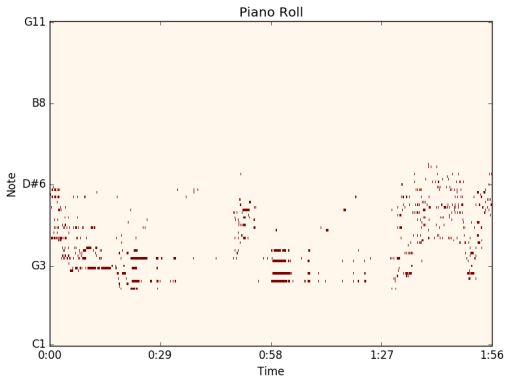
constant thresholding to produce a binary output which can be compared against the ground truth. However, constant thresholding has the disadvantage of being a manual process. It is necessary to select the threshold ahead of time and experiment to produce a better result. This project notes a significant loss in accuracy between the dataset during training and the postprocessed set.

The second problem lies in generalization. The overall dataset was split, 50% was selected for training, while 50% was left for testing. While the training data noted significant improvement in its score, this did not translate to the testing set as the results are much lower. This indicates a problem with generalization as new features are not being classified with the same accuracy as the features in the training set.

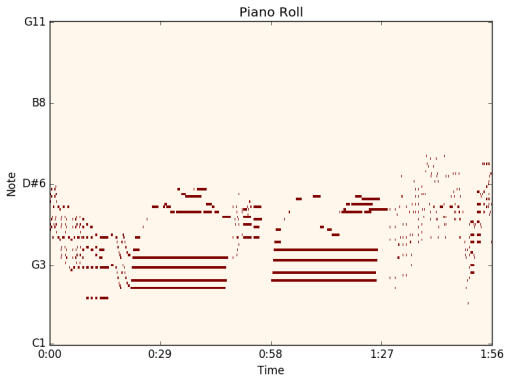
Finally, in the piano roll output, we observe that longer notes are fragmented and some offsets are not estimated properly. Overall, the ground truth and the output show the same general shape, however the Neural Network sometimes wrongly classifies transients as pitches. Also, some harmonic information is noteably missing.

6. CONCLUSIONS AND FUTURE WORK

In this report, we explore Neural Networks in their various forms and their applications in Automatic Music Transcription. In particular, we explore the use of Dense Neural Networks, Recurrent Neural Networks, and Convolutional Neural Networks in the literature. Finally, we implement our own transcription framework using a Dense Neural Network and examine its results.



(a) Piano Roll for the thresholded output



(b) Piano Roll for the ground truth

The most involved aspect of this project was in the preprocessing stage. The nature of the input and output datasets are not suited for processing using machine learning algorithms and much of this project's effort was spent in transforming the data both efficiently and accurately. While this was mostly accomplished in the early stages of the project, small adjustments needed to be made constantly. The transformation of audio-midi pairs into computable datasets was a challenge in itself and I believe that more work can be done at the preprocessing stage to produce more effective data to work with.

There are some noticeable limitations with the current framework. The most pressing issue is in the use of post processing. This framework only implements constant thresholding which offers very limited results. Since the acoustic model itself outputs non-binary values, some kind of thresholding is necessary in order to separate the most likely candidates from the features which scored too low. It's apparent however that this thresholding system needs to be more sophisticated than a simple cutoff as it was very difficult to determine the correct ratio which successfully ignored transients without removing correct, but lower scoring features. In [19], a Hidden Markov Model was used. This will be implemented as future work.

While the Dense Neural Network was mostly successful, there were other types of Neural Networks discussed in this report that couldn't be implemented due to time constraints. In the future, a Convolutional Neural Network

will be implemented to compare against the current framework. Furthermore, the framework allows for models to be built layer by layer, so more complex structures may be considered.

Another limitation with this framework is in the size of the dataset. It was observed during the testing of the framework that the score was raised drastically when more data was fed to the Neural Network. I believe that a larger dataset would contribute significantly to attaining a more accurate model. Unfortunately, it is difficult to obtain clean audio-midi pairs that are easy to work with.

7. REFERENCES

[1] Abdel-Hamid, Ossama et. al. "Convolutional Neural Networks for Speech Recognition." *IEEE/ACM Trans. Speech Audio Process.*, vol. 22, pp. 1533-1545, October 2014. https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/CNN_ASLPTrans2-14.pdf (accessed October 26th, 2016).

[2] Brian McFee, Matt McVicar, Colin Raffel, Dawen Liang, Oriol Nieto, Eric Battenberg, Josh Moore, Dan Ellis, Ryuichi Yamamoto, Rachel Bittner, Douglas Repetto, Petr Viktorin, João Felipe Santos, and Adrian Holovaty, "librosa: 0.4.1." Zenodo, 17-Oct-2015.

[3] Britz, Denny. "Understanding Convolutional Neural Networks For NLP." November 7, 2015. http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/ (accessed October 18th, 2016).

[4] Chollet, François, Keras (2015), GitHub, https://github.com/fchollet/keras

[5] Colin Raffel and Daniel P. W. Ellis. "Intuitive Analysis, Creation and Manipulation of MIDI Data with pretty midi." In 15th International Conference on Music Information Retrieval Late Breaking and Demo Papers, 2014.

[6] Convolutional Neural Networks (LeNet). http://deeplearning.net/tutorial/lenet.html (accessed October 5th, 2016).

[7] Deep Belief Networks. http://deeplearning.net/tutorial/DBN.html (accessed October 3rd, 2016).

[8] Dozat, Timothy. "Incorporating Nesterov Momentum into Adam." *Keras, Stanford.* http://cs229.stanford.edu/proj2015/054_report.pdf

[9] G. E. Poliner and D. P. Ellis, "A discriminative model for polyphonic piano transcription," *EURASHIP J. Appl. Signal Process.*, vol. 207 no. 1, pp. 154-154, 2007. https://www.ee.columbia.edu/~dpwe/pubs/PoliE06-piano.pdf (Accessed October 28th, 2016).

[10] Ghahramani, Z. (2001) "An Introduction to Hidden Markov Models and Bayesian Networks." *International Journal of Pattern Recognition and Artificial Intelligence.* 15(1):9-42.

[11] H. Larochelle and I. Murray, "The neural autoregressive distribution estimator," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2011, pp. 29-37.

[12] Jones E, Oliphant E, Peterson P, et al. "SciPy: Open Source Scientific Tools for Python," 2001-, http://www.scipy.org/ (Accessed October 4th, 2016).

[13] Kingma, Diederik "Adam: A Method for Stochastic Optimization." *eprint arXiv:1412.6980* (2014), https://arxiv.org/abs/1412.6980v8 (Acessed December 17th, 2016)

[14] Larochelle, Hugo. Neural networks [9.3]: Computer vision – parameter sharing. (Video) November 15, 2013. https://www.youtube.com/watch?v=aAT1t9p7ShM (accessed October 4th, 2016).

[15] Meinard Müller, Verena Konz, Wolfgang Bogler, and Vlora Arifi-Müller Saarland Music Data (SMD) In Late-Breaking and Demo Session of the International Conference on Music Information Retrieval (ISMIR), 2011.

[16] P. Smaragdis and J. C. Brown, "Non-negative matrix factorization for polyphonic music transcription," *in Proc. IEEE Workshop APPll. Signal Process. Audio Acoust.*, 2003, pp. 177-180. http://www.ee.columbia.edu/~dpwe/e6820/papers/SmarB03-nmf.pdf (Accessed October 28th, 2016).

[17] Rohrer, Brandon. How Convolutional Neural Networks work. August 18th, 2016. https://www.youtube.com/watch?v=FmpDIaiMIeA (accessed October 14th, 2016).

[18] Rojas Raul. "Neural Networks – A Systematic Introduction." *Springer-Verlag, Berline, New-York,* 1996. https://page.mi.fu-berlin.de/rojas/neural/ (accessed October 6th, 2016).

[19] Siddharth Sigtia, Emmanuiol Benetos, and Simon Dixon "An End-to-End Neural Network for Polyphonic Piano Music Transcription," in *Proc. IEEE Workshop APPll. Signal Process. Audio Acoust.*, vol. 24, no. 5, pp. 927-940, May 2016.

[20] Simpson, Andrew J.R. "Deep Karaoke: Extracting Vocals from Musical Mixtures Using a Convolutional Deep Neural Network." April 17, 2015. https://arxiv.org/ftp/arxiv/papers/1504/1504.04658.pdf (accessed September 28th, 2016).

[21] Srivasta Nitish, Geoffrey Hinton et al. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting.” *Journal of Machine Learning Research (2014).* Published April 2014. https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf (accessed October 4th, 2016).

[22] Welch, Steven. Neural Networks Demystified. (Video) November 4th, 2014. https://www.youtube.com/watch?v=bxe2T-V8XRs (accessed October 2nd, 2016).

[23] Welch, Steven. Neural Networks Demystified. (Code) Latest Commit August 23, 2016. https://github.com/stephencwelch/Neural-Networks-Demystified (accessed October 2nd, 2016).

[24] Weston, Jason. “Support Vector Machine (and Statistical Learning Theory) Tutorial.” Last Modified 2004. http://www.cs.columbia.edu/~kathy/cs4701/documents/jason_svm_tutorial.pdf (accessed October 6th, 2016).

[25] Zeiler, Matthew D. “ADADELTA: An Adaptive Learning Rate Model.” *eprint arXiv:1212.5701* (2012)